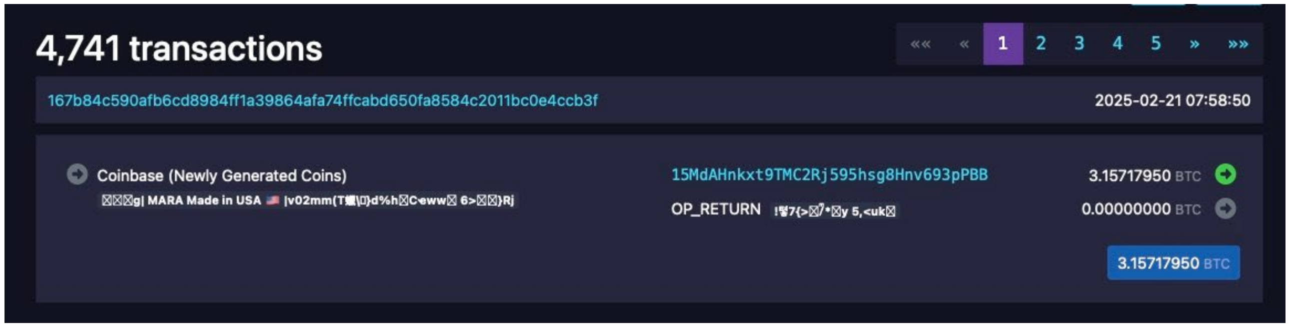
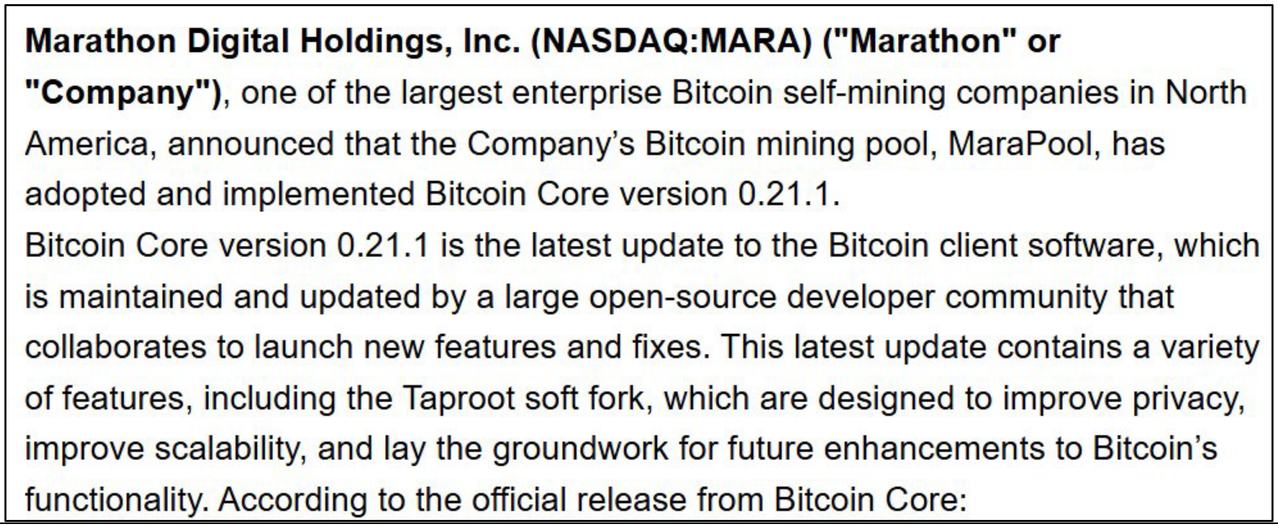
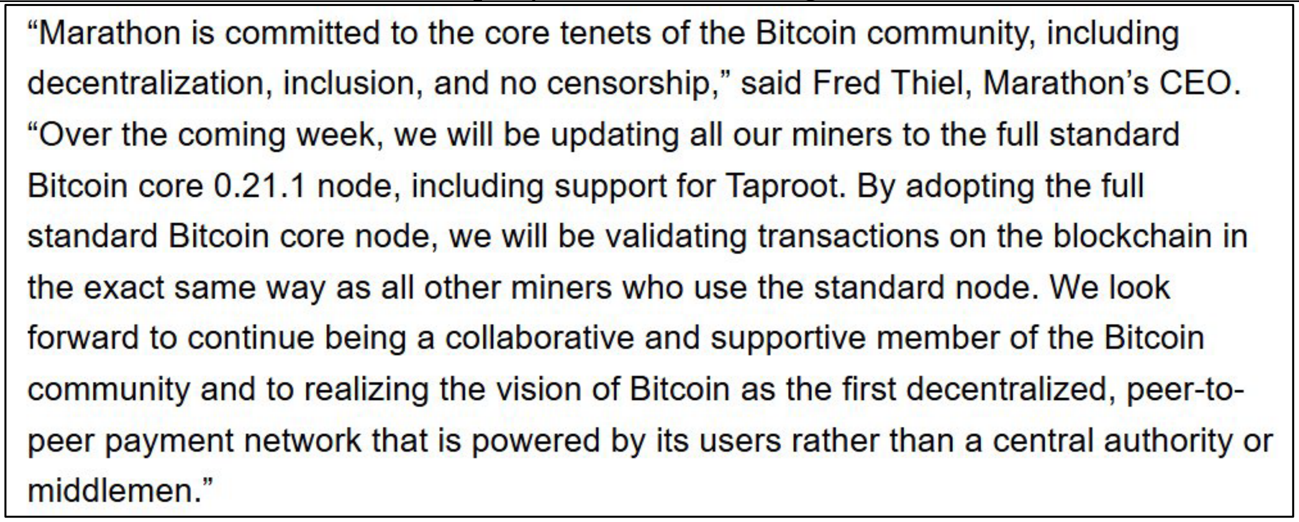
Exhibit 11

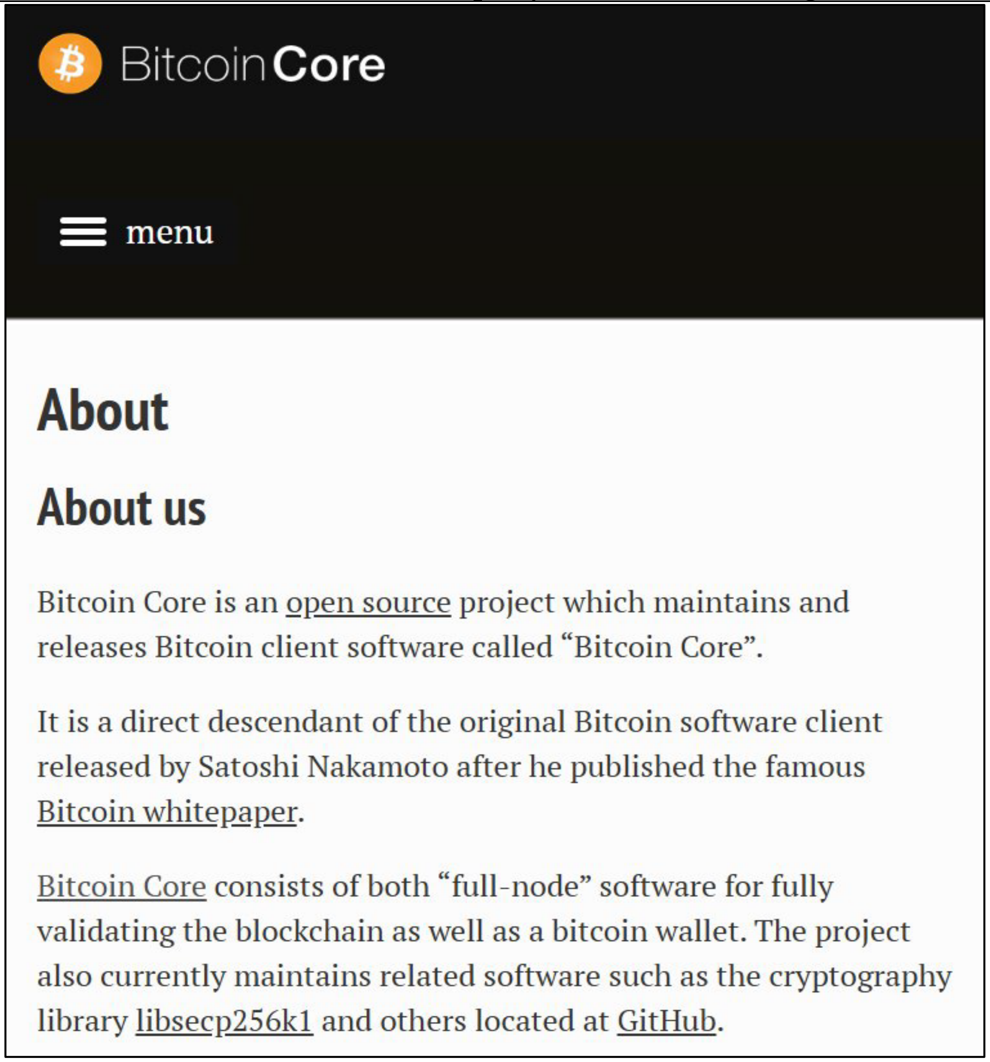
Exhibit 11: U.S. Patent No. 7,372,961

Claim 1	Exemplary Evidence of Infringement
[1pre] A method of generating a key k for use in a cryptographic function performed over a group of order q, said method including the steps of: … **[1g]** if said output H(SV) is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said output H(SV).	MARA Holdings, Inc. (hereinafter "MARA") performs a method for generating a key k for use in a cryptographic function performed over a group of order q, during the transfer of a Bitcoin to an address. *See*, *e.g.*: "Marathon is a digital asset technology company that is principally engaged in producing or **‘mining’ digital assets with a focus on the Bitcoin ecosystem** … **The term ‘Bitcoin’ with a capital ‘B’ is used to denote the Bitcoin protocol** which implements a highly available, public, permanent, and decentralized ledger." (Emphasis added) *See*, *e.g.*, MARA Holdings, Inc., Annual report pursuant to Section 13 and 15(d), (Form 10-K/A), at F-9, filed May 24, 2024, available at https://ir.mara.com/sec-filings/all-sec-filings/content/0001628280-24-025261/mara-20231231.htm. "The Bitcoin protocol is the technology that enables Bitcoin to function as a decentralized, peer-to-peer payment network. This open-source software, which sets the rules and processes that govern the Bitcoin network, is maintained and improved by a community of developers around the world known as Bitcoin Core developers … ‘At Marathon, we have historically focused on supporting Bitcoin by adding hash rate, which helps secure the network, and now, we are supporting those who maintain **the open-source protocol on which we all depend** by contributing to Brink,’ said Fred Thiel, Marathon's chairman and CEO." (Emphasis added) *See*, *e.g.*, Marathon Holdings Collaborates with Brink To Raise Up to $1 Million To Support Bitcoin Core Developers, GlobeNewswire (May 18, 2023), available at https://www.globenewswire.com/news-release/2023/05/18/2672276/0/en/Marathon-Digital-Holdings-Collaborates-with-Brink-To-Raise-Up-to-1-Million-To-Support-Bitcoin-Core-Developers.html. "The MaraPool wallet (Owned by the Company as Operator) is recorded on the distributed ledger as the winner of proof-of-work block rewards and assignee of all validations and, therefore, the transaction verifier of record. The pool participants entered into contracts with the Company as Operator; they did not directly enter into contracts with the network or the requester and were not

Claim 1	Exemplary Evidence of Infringement
	known verifiers of the transactions assigned to the pool…Therefore, the Company determined that it controlled the service of providing transaction verification services to the network and requester. **Accordingly, the Company recorded all of the transaction fees and block rewards earned from transactions assigned to the MaraPool as revenue, and the portion of the transaction fees and block rewards remitted to the MaraPool participants as cost of revenues**.” (Emphasis added). *See*, *e.g.*, MARA Holdings., Inc., Quarterly report, (Form 10-Q), at Note 4 – Revenues, filed November 12, 2024, available at https://www.sec.gov/ix?doc=/Archives/edgar/data/0001507605/000162828024047148/mara-20240930.htm.  4,741 transactions «« « 1 2 3 4 5 » »» 167b84c590afb6cd8984ff1a39864afa74ffcabd650fa8584c2011bc0e4ccb3f 2025-02-21 07:58:50 Coinbase (Newly Generated Coins) ☒☒☒g| MARA Made in USA 🇺🇸 |v02mm(T☒\☐}d%h☒C·eww☒ 6>☒☒}Rj 15MdAHnkxt9TMC2Rj595hsg8Hnv693pPBB 3.15717950 BTC OP_RETURN !☒7{>☒7•☒y 5,<uk☒ 0.00000000 BTC 3.15717950 BTC *See*, *e.g.*, https://mempool.space/address/15MdAHnkxt9TMC2Rj595hsg8Hnv693pPBB. “**Bitcoin signed messages have three parts, which are the Message, Address, and Signature**. The message is the actual message text - all kinds of text is supported, but it is recommended to avoid using non-ASCII characters in the signature because they might be encoded in different character sets, preventing signature verification from succeeding. The address is a legacy, nested segwit, or native segwit address. Message signing from legacy addresses was added by Satoshi himself and therefore does not have a BIP. **Message signing from segwit addresses has been added by BIP137 … The Signature is a base64-encoded ECDSA signature** that, when decoded, with fields described in the next section.” (Emphasis added) *See*, *e.g.,* Message Signing, https://en.bitcoin.it/wiki/Message_signing.

Claim 1	Exemplary Evidence of Infringement
	"This document describes a signature format for **signing messages with Bitcoin private keys**. The specification is intended to describe the standard for signatures of messages that can be signed and verified between different clients that exist in the field today." (Emphasis added) *See*, *e.g.*, Bitcoin BIP137, https://github.com/bitcoin/bips/blob/master/bip-0137.mediawiki. For example, MARA uses Bitcoin Core for generating a key k. *See*, *e.g.*:  **Marathon Digital Holdings, Inc. (NASDAQ:MARA) ("Marathon" or "Company")**, one of the largest enterprise Bitcoin self-mining companies in North America, announced that the Company's Bitcoin mining pool, MaraPool, has adopted and implemented Bitcoin Core version 0.21.1. Bitcoin Core version 0.21.1 is the latest update to the Bitcoin client software, which is maintained and updated by a large open-source developer community that collaborates to launch new features and fixes. This latest update contains a variety of features, including the Taproot soft fork, which are designed to improve privacy, improve scalability, and lay the groundwork for future enhancements to Bitcoin's functionality. According to the official release from Bitcoin Core:

Claim 1	Exemplary Evidence of Infringement
	 "Marathon is committed to the core tenets of the Bitcoin community, including decentralization, inclusion, and no censorship," said Fred Thiel, Marathon's CEO. "Over the coming week, we will be updating all our miners to the full standard Bitcoin core 0.21.1 node, including support for Taproot. By adopting the full standard Bitcoin core node, we will be validating transactions on the blockchain in the exact same way as all other miners who use the standard node. We look forward to continue being a collaborative and supportive member of the Bitcoin community and to realizing the vision of Bitcoin as the first decentralized, peer-to-peer payment network that is powered by its users rather than a central authority or middlemen." *See*, *e.g.*, https://br.advfn.com/bolsa-de-valores/nasdaq/MARA/share-news/85244958/marathon-signals-for-taproot.

Claim 1	Exemplary Evidence of Infringement
	 About About us Bitcoin Core is an open source project which maintains and releases Bitcoin client software called “Bitcoin Core”. It is a direct descendant of the original Bitcoin software client released by Satoshi Nakamoto after he published the famous Bitcoin whitepaper. Bitcoin Core consists of both “full-node” software for fully validating the blockchain as well as a bitcoin wallet. The project also currently maintains related software such as the cryptography library libsecp256k1 and others located at GitHub. *See*, *e.g.*, Bitcoin core, https://github.com/bitcoin/bitcoin; *see also* https://github.com/bitmaintech/cgminer

Claim 1	Exemplary Evidence of Infringement

[1pre] A method of generating a key k for use in a cryptographic function performed …:
 	
[1a] generating a seed value SV from a random number generator;
 	
[1b] performing a hash function H( ) on said seed value SV to provide an output H(SV);
 	
[1c] determining …;
 	
[1d] accepting said output H(SV) …;
 	
[1e] rejecting said output H(SV) …;
 	
[1f] if said output H(SV) is rejected, repeating said method; and
 	
[1g] if said output H(SV) is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said output H(SV).
	MARA uses function `MakeNewKey` to generate a private key k for use in a cryptographic function, as evidenced by the Bitcoin Core code below.
	
	`bool CKey::Check(const unsigned char *vch) {`
	`    return secp256k1_ec_seckey_verify(secp256k1_context_sign, vch);`
	`}`
	
	`void CKey::MakeNewKey(bool fCompressedIn) {`
	`    MakeKeyData();`
	`    do {`
	`        GetStrongRandBytes(*keydata);`
	`    } while (!Check(keydata->data()));`
	`    fCompressed = fCompressedIn;`
	`}`
	
	See, *e.g.*, bitcoin\src\key.cpp
	
	`class RNGState {`
	`    Mutex m_mutex;`
	`    /* ... To protect against situations where an attacker might`
	`     * observe the RNG's state, fresh entropy is always mixed when`
	`     * GetStrongRandBytes is called.`
	`     */`
	`    ...;`
	`}`
	
	`void GetStrongRandBytes(Span<unsigned char> bytes) noexcept`
	`{`
	`    ProcRand(bytes.data(), bytes.size(), RNGLevel::SLOW, /*always_use_real_rng=*/true);`
	`}`
	
	See, *e.g.*, bitcoin/src/random.cpp

Claim 1	Exemplary Evidence of Infringement

[1pre] A method of generating a key k for use in a cryptographic function performed over a group of order q, said method including the steps of:
 	
[1a] generating a seed value SV from a random number generator;
 	
[1b] performing a hash function H( ) on said seed value SV to provide an output H(SV);
 	
…
	MARA uses class `CSHA512` to hash the random number generator state and outputs H(SV).
	
	`void ProcRand(unsigned char* out, int num, RNGLevel level, bool always_use_real_rng)`
	`noexcept {`
	`    if (!rng.MixExtract(out, num, std::move(hasher), false, always_use_real_rng)) { ...`
	`    }`
	`}`
	
	`/** Extract up to 32 bytes of entropy from the RNG state, mixing in new entropy`
	` * from hasher. ...`
	` */`
	`bool MixExtract(unsigned char* out, size_t num, CSHA512&& hasher, bool strong_seed,`
	`  bool always_use_real_rng) noexcept EXCLUSIVE_LOCKS_REQUIRED(!m_mutex) { ...`
	`    {`
	`        ...;`
	`        // Write the current state of the RNG... a new counter ... into the`
	`state/hasher`
	`        hasher.Write(...);`
	`        ...;`
	`        hasher.Finalize(buf);`
	`        ...;`
	`        memcpy(out, buf, num);`
	`    } ...;`
	`}`
	See, *e.g.*, bitcoin/src/random.cpp
	
	`/** A hasher class for SHA-512. */`
	`class CSHA512`
	`{ ... };`
	See, *e.g.*, bitcoin/src/crypto/sha512.h

[1pre] A method of generating a key k for use in a cryptographic function performed over a group of order q, said method including the steps of:	Elliptic curve secp256k1 is a group of order q. Rather than performing a modulo function – and introducing a bias – the output of H() is evaluated.
	
	`bool CKey::Check(const unsigned char *vch) {`
	`    return secp256k1_ec_seckey_verify(secp256k1_context_sign, vch);`
	`}`

Claim 1	Exemplary Evidence of Infringement

…
 	
[1c] determining whether said output H(SV) is less than said order q prior to reducing mod q;
 	
[1d] accepting said output H(SV) for use as said key k if the value of said output H(SV) is less than said order q;
 	
[1e] rejecting said output H(SV) as said key if said value is not less than said order q;
 	
…
	See, *e.g.*, bitcoin\src\key.cpp
	
	`/** Verify an elliptic curve secret key.`
	` *`
	` * A secret key is valid if it is not 0 and less than the secp256k1 curve order`
	` * when interpreted as an integer (most significant byte first). ...`
	` * ...`
	` * Returns: 1: secret key is valid`
	` * 0: secret key is invalid`
	` * Args: ctx: pointer to a context object.`
	` * In: seckey: pointer to a 32-byte secret key.`
	` */`
	`SECP256K1_API SECP256K1_WARN_UNUSED_RESULT int secp256k1_ec_seckey_verify(`
	`    const secp256k1_context *ctx, const unsigned char *seckey) ...;`
	
	See, *e.g.*, bitcoin/src/secp256k1/include/secp256k1.h

[1pre] A method of generating a key k for use in a cryptographic function performed …:
 	
…
 	
[1b] performing a hash function H( ) on said seed value SV to provide an output H(SV);
	MARA uses function `MakeNewKey` to generate a private key k.
	
	`bool CKey::Check(const unsigned char *vch) {`
	`    return secp256k1_ec_seckey_verify(secp256k1_context_sign, vch);`
	`}`
	
	`void CKey::MakeNewKey(bool fCompressedIn) {`
	`    MakeKeyData();`
	`    do {`
	`        GetStrongRandBytes(*keydata);`
	`    } while (!Check(keydata->data()));`
	`    fCompressed = fCompressedIn;`
	`}`
	
	See, *e.g.*, bitcoin\src\key.cpp

Claim 1	Exemplary Evidence of Infringement
[1c] determining …; **[1d]** accepting said output H(SV) …; **[1e]** rejecting said output H(SV) …; **[1f]** if said output H(SV) is rejected, repeating said method; and **[1g]** if said output H(SV) is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said output H(SV).	`** An encapsulated` **`private key`**`. */` `class` **`CKey`** `{` `...:` `private:` `/** Internal data container for private key material. */` `using KeyType = std::array<unsigned char, 32>;` `...[` `//! The actual byte data. nullptr for invalid keys.` `secure_unique_ptr<KeyType>` **`keydata`**`;` `...;` `};` *See*, *e.g.*, bitcoin/src/key.h